IOP Conf. Series: Materials Science and Engineering **263** (2017) 042138 doi:10.1088/1757-899X/263/4/042138

Sensitivity analysis of linear programming problem through a recurrent neural network

Raja Das
Department of Mathematics, School of Advanced Sciences, VIT University, Vellore-632014, India

E-mail: rdasresearch@gmail.com

Abstract. In this paper we study the recurrent neural network for solving linear programming problems. To achieve optimality in accuracy and also in computational effort, an algorithm is presented. We investigate the sensitivity analysis of linear programming problem through the neural network. A detailed example is also presented to demonstrate the performance of the recurrent neural network.

1. Introduction

In the last 50 years, researchers have proposed various dynamic solvers for solving linear programming problems. The dynamical systems approach to solving constrained optimization problems was first proposed by Pyne[1]. Recently, due to renewed interest in neural networks, several neural network based dynamic solvers have been proposed-see e.g., [2,3,4]. Recently, various researches have been advanced for the application of the technology related to knowledge engineering such as ANN to the engineering field. Mathematical optimization problems have been widely applied in practically all knowledge areas. Constrained linear optimization plays a fundamental role in many problems involving with the areas of science and engineering, where a set of design parameters is optimized subject to inequality and / or equality constrained. We are interested in linear programming problems since they are one of the most frequently used constrained optimization problems. Linear programming has been widely applied in practically every area of production, economic, social and government planning.

In this paper, we study the design of the recurrent neural network and the sensitivity analysis of linear programming problem through it. In 1947 Dantzig[5] developed a method for solving linear programming problems which is known today as the Simplex method. Brown and Koopmans [6] described the first of a series of interior point methods for solving linear programs. Recently, Karmarkar[7] developed an algorithm which appears to be more efficient than the Simplex Method on some complicated real-world problems of scheduling, routing and planning. Conn. developed alternative methods for solving linear programs using unconstrained optimization combined with penalty function methods.

In the neural networks literature, there are several approaches to the solution of linear optimization problems. The first neural approach applied to optimization problems was proposed by Tank and Hopfield [2] where the network used for solving linear programming problems.

The Primary objective of this paper is to present the alternative technique for finding the range of the coefficients of the objective function and the range of the change in the right hand side constant of the linear programming problem. We review the terms related to the neural network for solving linear programming problem.

2. Problem statement

Consider a standard form of linear Programming Problems described as

Minimize $Z = C^T X$ (1)

subject to $AX = B$ (2)

and $X \geq 0$ (3)

where $X \in R^n$ is a column vector of decision variables. $C \in R^n$ and $B \in R^m$ are column vectors of cost coefficient and right hand side parameters, respectively, $A \in R^{m \times n}$ is a constraint coefficient matrix, and the subscript *T* denotes the transpose operator.

3. Solution to the LP problem

In general, the LP problem can have four possible solution type:

1. *Unique Solution*. There is only one solution that satisfies all constraints, and the objective function reaches a minimum within the feasible region.

2. *Nonunique Solution*. There are several feasible solutions where the objective function reaches a minimum.

3. *An unbounded solution*. The objective function is not bounded in the feasible region and it approaches ∞.

4. *No feasible solution*. Constraint provided in (2) and (3) are too restrictive, and the set of feasible solution is empty.

Although theoretically valid, cases 3 and 4 appear rarely in engineering and scientific applications. Furthermore, they can be easily detected, and in further consideration of the LP problem we will assume that it is formulated in such a way that there exists at least one feasible solution.

4. The neural network

An artificial neural network (ANN) is a dynamic system, consisting of highly interconnected and parallel non-linear processing elements, that is highly efficient in computation. In this paper, a recurrent network[7] with equilibrium points representing a solution of the constrained optimization problem has been developed. As introduced in Hopfield [2] these network are composed with feedback connection between nodes. In the standard case, the nodes are fully connected i,e. every node is connected to all other nodes, including itself. The node equation for the continuous-time network with *n*- neurons is given by

$$x_i(k+1) = \begin{cases} x_i(k) - \mu\left[c_i + \sum a_{ij}\lambda_j(k)\right], & \text{if } x_i(k+1) > 0 \\ 0, & \text{if } x_i(k+1) \leq 0 \end{cases}$$

and

$$\lambda_j(k+1) = \lambda_j(k) + \eta\{r_j(k) - \alpha\lambda_j(k)\}$$

where

$$r_j(k) = AX - B$$

The first step in a neural network implementation for solving the LP problem is to define an energy function that can be optimized in an unconstrained fashion. To accomplish this, the linear constraints $AX = B$ and non-negativity constraints $X > 0$ are appended to the objective function in some convenient way.

Commonly the constraints are incorporated as penalty terms that, whenever violated, increase the value of the energy function. Two energy function that can be derived using the Lagrange multiplier method are defined as

$$E_1(X) = L_1(X,\lambda) = C^T X + \frac{K}{2}(AX-B)^T(AX-B) + \lambda^T(AX-B)$$

$$E_2(X) = L_2(X,\lambda) = C^T X + \frac{K}{2}(AX-B)^T(AX-B) + \lambda^T(AX-B) - \alpha\lambda^T\lambda \quad (4)$$

with K and $\alpha \geq 0$, $R^{m\times 1}$, and $X \geq 0$.

Appling the method steepest descent in discrete time, we compute the gradient of the energy function in (4) with respect to X and obtain

$$\nabla_X E_2 = \frac{\partial E_2(X,\lambda)}{\partial X} = \frac{\partial}{\partial X}\left\{C^T X + \frac{K}{2}\left(X^T A^T AX - X^T A^T B - B^T AX\right) + B^T B + \lambda^T(AX-B) - \alpha\lambda^T\lambda\right\}$$

and

$$\frac{\partial E_2}{\partial X} = C + KA^T(AX-B) + A^T\lambda = C + A^T(Kr+\lambda)$$

Where $r \in R^{m\times 1}$ is defined as

$$r = r(k) = AX(k) - B$$

In a similar manner we have

$$\frac{\partial E_2}{\partial \lambda} = AX - B - \alpha\lambda = r - \alpha\lambda$$

5. Sensitivity analysis

The optimal solution of the linear programming problem (1), (2) & (3) depends upon the parameters (c_j, a_{ij} and b_i) of the problem. If the optimal value of the objective function is relative sensitive to changes in certain parameters, special care should be taken in estimating these parameters and in selecting a solution which does well for most of their likely values. Then, it is quite important to know the range of the cost for which the solution remains optimal.

The investigation that deals with changes in the optimal solution due to changes in the parameters (c_j, a_{ij} and b_i) is called sensitivity analysis.

The changes in the linear programming problem which are usually studied by sensitivity analysis include:

1. Coefficients (c_j) of the objective function.
2. Change in the right hand side (b_j) constants.

Lemma.If $d_j \geq 0$, $d_j + \rho d'_j \geq 0$, $j = 1,2,\cdots,n$ then $\underset{j}{Max}_{(d'_j>0)}\left[-\frac{d_j}{d'_j}\right] \leq \rho \leq \underset{j}{Min}_{(d'_j<0)}\left[-\frac{d_j}{d'_j}\right]$.

The range of cost for which the solution remains optimal can be determined using the above lemma. But the developed recurrent neural network may not give the range of the coefficient (c_j) of the objective function. In this case, the primal form of the linear programming has to be written in the dual form. We can determine the range of the change in the right hand side constant (b_j) through the

proposed recurrent neural network. The range of the right hand side constant (b_j) of dual problem is same as the range of coefficient c_j of the objective function of the primal problem.

6. Numerical Examples

First, we give one numerical example to demonstrate the sensitivity analysis of linear programming problem through the proposed Hopfield neural network.

Example 1: Consider the linear programming problem

$$\text{Max } Z = 45x_1 + 80x_2$$

Subject to

$$\begin{aligned} 5x_1 + 20x_2 &\le 400 \\ 10x_1 + 15x_2 &\le 450 \\ x_1,\ x_2 &\ge 0 \end{aligned} \qquad \text{(A)}$$

From the problem statement it can be seen that the linear programming problem is not in the standard form. By adding surplus variables x_3 and x_4, the linear programming problem can be transferred in the standard form as follows:

$$\text{Max } Z = 45x_1 + 80x_2 + 0x_3 + 0x_4$$

Subject to

$$\begin{aligned} 5x_1 + 20x_2 + x_3 &= 400 \\ 10x_1 + 15x_2 + x_4 &= 450 \\ x_1,\ x_2, x_3, x_4 &\ge 0 \end{aligned} \qquad \text{(B)}$$

To solve this linear programming problem, the neural network in Figure 1 is simulated. The parameters of the network were chosen as $\mu = 0.01, \eta = 0.01$. Zero initial conditions were assumed both for x and λ. The network converges in approximately 4,000 iterations. The solution to the linear programming problem is given as $\tilde{x}^* = [23.34, 24.24, 14]^T$, which is within the learning rate parameter accuracy from the exact solution $\tilde{x} = [23, 14]^T$

Table 1. Comparison of the simulation result with the exact solution of the example.

Variables	Neural Network Result	Exact Solution
x_1	23.34	24
x_2	14.24	14

The results are also given in Table 1 and they are very close to the exact solution. The trajectories of the neural network variables are plotted in Figure 2.

This proposed recurrent neural network could not give the range of the coefficient of the objective function of linear programming problem (A). But this network determined the range of the right hand side constants through this developed network. The range is given in the Table 2.

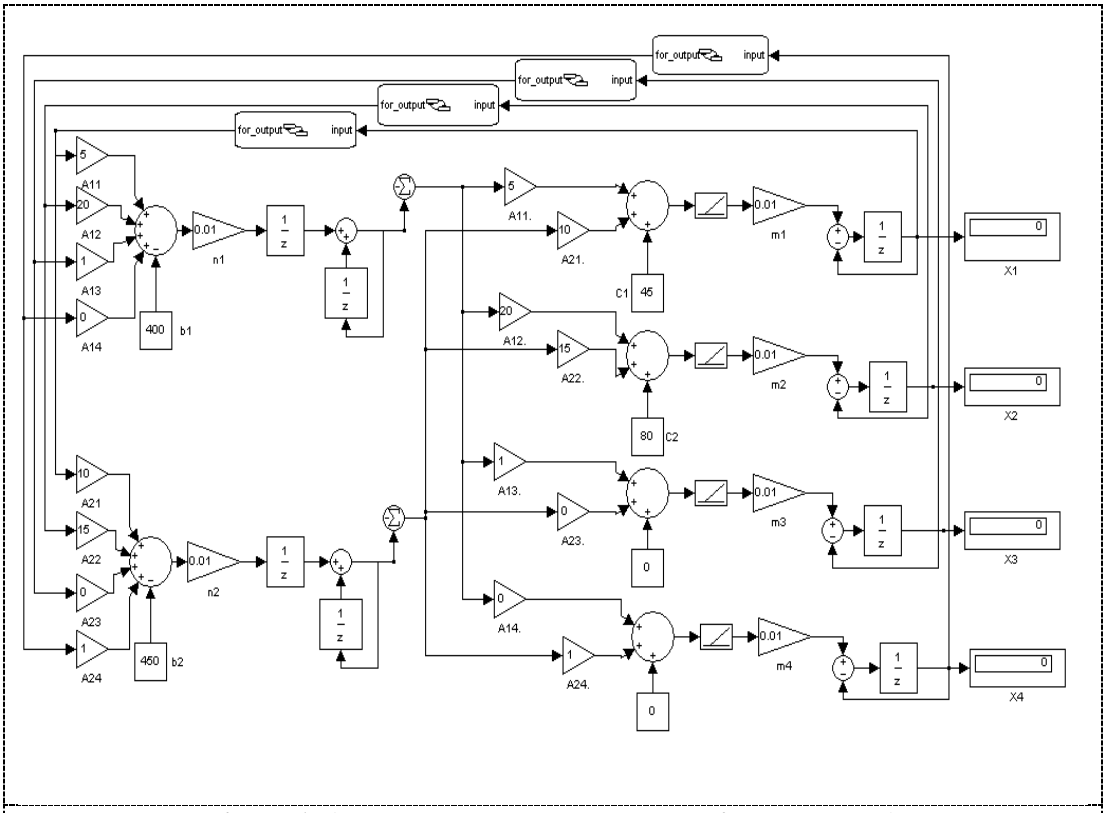

Figure 1. (Recurrent Neural Network Primal Problem)

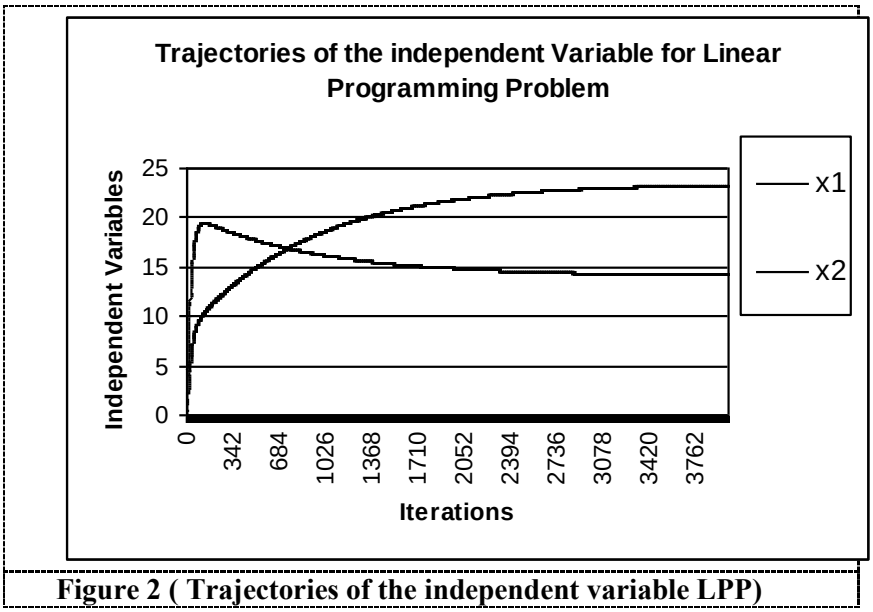

Figure 2 (Trajectories of the independent variable LPP)

Table 2. Comparison of the simulation result with the exact range of b_j

b_j	Neural Network	Actual
b_1	$221.5 \le b_1 \le 602.2$	$225 \le b_1 \le 600$

b_2	$298.7 \le b_2 \le 813$	$300 \le b_2 \le 800$

7. Conclusion

We investigated in this paper the MATLAB Simulink modeling and simulative confirmation of such a recurrent neural network. It converges to the actual solutions of the Linear Programming Problem. We have concluded that the recurrent neural network can be used for determining the range of the coefficient of the objective function and the right hand side of the problem by using the concept of duality in linear programming problem.

References

[1] Pyne I B 1956 *Trans. Amer. Inst.Eng.,* **75** 139-143.
[2] Tank D W and Hopfield J J 1986 *IEEE Trans. Circ. Syst*., **33** 533-541
[3] Kennedy M P and Chua L O 1988 *IEEE Trans. Circ. Syst.,* **35** 554-562
[4] Lillo W E, Loh M H, Hui S and Zak S H 1991 *Tech. rep. TR-EE* 91-43, Purdue Univ., W. Lafayette IN. Oct.
[5] Dantzig G B 1963Linear Programming and Extesions. Princeton. NJ: Princeton Press.
[6] Brown G W and Koopmans T C 1951 Computation suggestions for maximizing a linear function subject to linear equalities," in Activity Analysis of Production and Allocation, T. Koopmans. Ed. New York: J Wiley, 377-380
[7] Fredric M. Ham and Ivica Kostanic 2002Principles of Neurocomputing for Science and Engineering" Tata McGraw-Hill.
[8] Gass S I 1984Linear Programming, Methods and Applications, 5th ed., New York: McGraw-Hill.
[9] Belegundu A D and Chandrupatla T R 2003 Optimization Concept and Applications in Engineering" Pearson Education